Udacity Data Engineering Nanodegree Capstone Project

John Maxi

**Step 1: Scope the Project and Gather Data**

The goal of my project is to build an analytics table for researching the relationship between air quality and person-level health outcomes. Air quality data will be obtained from the EPA <https://aqs.epa.gov/aqsweb/airdata/download_files.html> and person-level health outcome will be obtained from the Nation Health Interview Survey <https://www.cdc.gov/nchs/nhis/data-questionnaires-documentation.htm>. I have collected data from 2014 onwards for the EPA data, and 2015 onwards for the NHIS data. In total, the raw data contains 10s of millions of rows.

The goal of the data model is to be able to explore spatial-temporal relationships between the air quality and the health outcomes. The data model will need to have the capacity to marry the time and location of the air quality measurements with the time and location of the health outcomes data.

**Step 2: Explore and Assess the Data**

The data exploration can be viewed in detail in the file: Step 2 - EDA and Data Cleaning. To clean the data I took the following steps for each dataset:

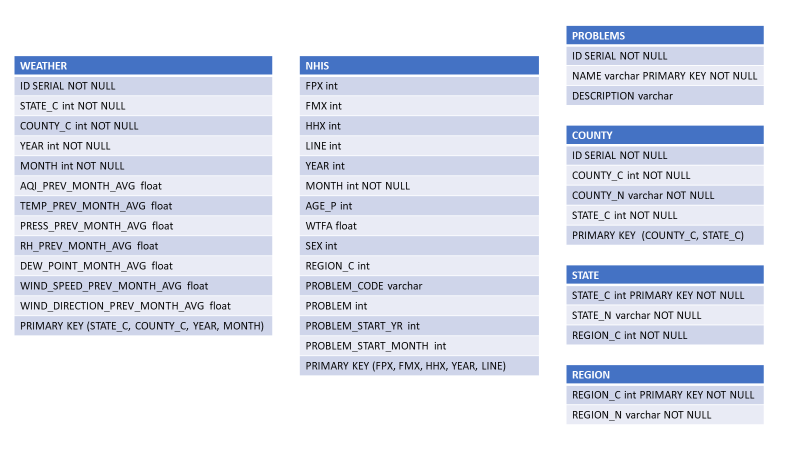
Weather data:

1. Open and concatenate each type of weather variable file into a single table.
2. Filter the PM 2.5 rows so as to keep only the rows that are measured in the accepted standard.
3. Convert date strings to datetimes for processing.
4. Extract the unique state code and names in the dataset and map the states to their respective regions based on: <https://www.businessinsider.com/regions-of-united-states-2018-5>. The only geographic information available in the NHIS dataset is the region of the country. Creating the map from state to region is necessary to overlap the locations of the sensors and health outcomes. The granularity of the NHIS data necessitates that the final analysis will cover broad swaths of the country. States that are mapped to one of the four regions, e.g., Puerto Rico, their region value is set to -1.
5. The monthly average values at each location are calculated for each of the weather variables (air quality, humidity, dew point, temperature, wind speed, wind direction, pressure).
6. Next, a date offset of one month is added to the datetime value associated with the average weather variables. This causes the weather data to be one month in the past relative to its corresponding time stamp. This is performed because we want to be able to investigate how the past months weather affected health problems in the next month.
7. Finally, I have chosen to store the year and month associated with the readings as two columns representing the year and the month respectively, rather than using a datetime value. This is because the granularity of the NHIS data is only year and month. Storing more detailed time data would be a false representation of the data. (Similar to choosing the correct number of significant digits in science. We wouldn’t report the number of digits the calculator is capable of displaying, we would only display as many digits as our instruments are capable of measuring).

NHIS data:

1. Drop data rows that have been flagged as poor quality by NHIS.
2. Each of the different types of health problems recorded by the survey are given their own column in the raw data, along with columns describing how the long the problems have lasted. Additionally, problems for children are separate from problems for adults. For each combination of problem type, duration, and survey date, I calculate when the problem started relative to the month the survey occurred for that person.
3. A single person in the NHIS survey is represented by four variables: person number, family number, house hold number and survey year. Since I am processing the data in yearly data files, I use the first three variables to determine how many times a given entity occurs in the data. These values become the LINE value in the final data model.
4. Next, I extract the year and month the problem started as integer values like in the weather table.
5. In the raw data, all problems associated with children are distinct from problems associated with adults, even if the problems are similar (e.g., there is a vision problem code for adults, and separate one for children). I combine the similar problems into a single code. The end user will be left to filter on the age column if they want to separate out children from adults for these problems.
6. Finally, the problem codes are extracted from the raw data and mapped to their description. This becomes the problem dimension table in the final data model.

**Step 3: Define the Data Model**

 Figure 1: ER diagram of the data model.

The chosen data model allows for combining information from different sources based on time and location. There are two fact tables, WEATHER and NHIS. WEATHER holds the information from air quality sensors summarized by month and location. The values associated with each row (a time and location) indicate the average value for each measurement type from the previous month at that location. The year and month columns indicate the current month. In turn, year and month columns can be used to either the year and month a person was surveyed using the YEAR and MONTH in the NHIS fact table, or the time a person identified a problem as starting using the PROBLEM\_START\_YR and PROBLEM\_START\_MONTH columns. NHIS holds the responses of participants in the NHIS study regarding their medical problems. Each row represents a person’s response to a given type of health problem. NHIS includes the variable REGION\_C that can be used to connect the location of a person in the survey to the location of a sensor. Note that the sensor information must be rolled up to the region level using the appropriate dimension tables, COUNTY, STATE and/or REGION.

Pipeline steps:

1. Clone the git repo:
2. Download data from sources:
   1. Weather source: <https://aqs.epa.gov/aqsweb/airdata/download_files.html>
      1. Use the daily zip files for 2014 - 2018
   2. NHIS source: <https://www.cdc.gov/nchs/nhis/nhis_2015_data_release.htm>
      1. Use the person data zip files for 2015 - 2018
3. Setup project directory:
   1. Data should be in the data folder: 1 zip folder per year of data downloaded from above sources.
   2. Create a directory “logs” in the data folder
4. Clone anaconda env using environment.yml file. In the anaconda terminal:

conda env create -f environment.yml

1. Run tests.py to test the helper functions and the correct directory structure
2. Run python csv\_to\_json.py to convert some of the csv data to JSON (this is to meet project requirement of using different data formats, it is not necessary for recreating the project).
3. Run create\_tables.py
4. Run etl.py
5. Use the sample queries to get started with analysis!

**Step 4: Run ETL to Model the Data**

Data Dictionary:

WEATHER table: The weather fact table holds information about air quality and meteorological variables at distinct times and locations. The primary key is made up of STATE\_C, COUNTY\_C, YEAR, MONTH. These variables can be used to join the facts to location and time columns of other fact and dimension tables.

ID: the row number of the table, in the order in which new data is added.

STATE\_C: An integer value representing an American state

COUNTY\_C: An integer value representing a county within a given state. Counties in different states can have the same county code. County code within states are unique.

YEAR: An integer value representing a given year

MONTH: An integer value representing a given month

AQI\_PREV\_MONTH\_AVG: a numerical value representing the average AQI (PM 2.5-based air quality index) during the month prior to YEAR and MONTH at the COUNTY\_C/STATE\_C location. Note that if multiple sensors are present in a given state/county, they have been averaged together in this model.

TEMP\_PREV\_MONTH\_AVG: a numerical value representing the average temperature (in Fahrenheit) during the month prior to YEAR and MONTH at the COUNTY\_C/STATE\_C location. Note that if multiple sensors are present in a given state/county, they have been averaged together in this model.

PRESS\_PREV\_MONTH: a numerical value representing the average barometric pressure (in millibars) during the month prior to YEAR and MONTH at the COUNTY\_C/STATE\_C location. Note that if multiple sensors are present in a given state/county, they have been averaged together in this model.

RH\_PREV\_MONTH\_AVG: a numerical value representing the average relative humidity (as percentage) during the month prior to YEAR and MONTH at the COUNTY\_C/STATE\_C location. Note that if multiple sensors are present in a given state/county, they have been averaged together in this model.

DEW\_POINT\_PREV\_MONTH\_AVG: a numerical value representing the average dew point (in degrees Fahrenheit) during the month prior to YEAR and MONTH at the COUNTY\_C/STATE\_C location. Note that if multiple sensors are present in a given state/county, they have been averaged together in this model.

WIND\_SPEED\_PREV\_MONTH\_AVG: a numerical value representing the average wind speed (in knots) during the month prior to YEAR and MONTH at the COUNTY\_C/STATE\_C location. Note that if multiple sensors are present in a given state/county, they have been averaged together in this model.

WIND\_DIRECTION\_PREV\_MONTH\_AVG: a numerical value representing the average wind direction (degrees compass) during the month prior to YEAR and MONTH at the COUNTY\_C/STATE\_C location. Note that if multiple sensors are present in a given state/county, they have been averaged together in this model.

NHIS table: The NHIS fact table holds response to survey questions about health problems from the National Health Interview Survey. The primary key consists of FPX, FMX, HHX, YEAR, LINE and represents a single person/problem combination. The time and space columns YEAR, MONTH, PROBLEM\_START\_YR, PROBLEM\_START\_MON, and REGION\_C can be used to join the data to other fact and dimension tables.

FPX: integer representing a numbered person within a family. Used in combination with FMX, HHX, and YEAR to identify individual persons

FMX: integer representing a family number. Used in combination with HHX and YEAR to identify individual families.

HHX: integer representing a household. Used in combination with YEAR to identify individual households.

LINE: An integer representing a problem that an individual person may have. For each (FPX, FMX, HHX, YEAR) tuple, there are as many LINE values as there are potential or actual problems a person may have.

YEAR: An integer indicating what year the person was surveyed in.

MONTH: An integer indicating what month of the year the person was surveyed in.

AGE\_P: An integer indicating the age of the person at the time of the survey. Age values are capped at 85. Any one older than 85 is assigned the value of 85.

WTFA: Weight – Final Annual. A numeric value representing post-stratification adjustments using Census Bureau population control totals. Adjustments are made for demographic variables (such as age, race/ethnicity, sex). NHIS recommends using this weight for analysis at the person level when using a full year of data. The sum of the weights are equal to the average of the civilian, noninstitutionalized U.S. population estimates for February, May, August, and November.

SEX: Integer representing 1, male, and 2, female

REGION\_C: An integer representing which geographic region of the United States a person. 1: Northeast, 2: Midwest, 3: South, 4: West.

PROBLEM\_CODE: A string value indicating the type of problem. The description of the problem code is can be found by joining to PROBLEMS table on NAME column.

PROBLEM: 0 or 1, indicates if the person indicates they have that particular health problem.

PROBLEM\_START\_YR: The year the problem started. This value is calculated from variables collected in survey representing the number of time units in the past the respondent indicates as the time the problem started. The start date of the problem is estimated as: (SRVY\_YEAR + INTV\_MON) - (number of units \* unit). The year is then extracted from the resulting start date.

PROBLEM\_START\_MONTH: The month the problem started. Derived from the same calculation used for PROBLEM\_START\_YR, but extracting the month value from the start date.

PROBLEMS table:

ID: a serial integer representing the order in which data was added to the table.

NAME: a string representing the name given to the variable in the NHIS survey. Foreign key for NHIS.PROBLEM.

DESCRIPTION: a string describing the problem as I the NHIS documentation.

COUNTY table:

ID: a serial integer representing the order in which data was added to the table.

COUNTY\_C: An integer value representing a county within a given state. Counties in different states can have the same county code. County code within states are unique.

COUNTY\_N: A string, the name of the county.

STATE\_C: An integer value representing an American state

STATE table:

STATE\_C: An integer value representing an American state

STATE\_N: The name of the state

REGION\_C: The US region the state in which the state resides.

REGION table:

REGION\_C: Integer representing a region of the United States

REGION\_N: The name of each region. 1: Northeast, 2: Midwest, 3: South, 4: West.

**Data Quality Checks**

Data integrity of the pipeline is ensured by using appropriate primary keys, conflict handling during inserts, and logging. Dimension tables do not change when conflicting data is attempted to be inserted. These tables generally hold information that is not expected to change (state names, county names, regions of the country) nor would we want historical values to be overwritten (problem codes). The fact tables, WEATHER and NHIS, will update their values when data is inserted into an existing primary key. The primary key represents a place at a specific time (WEATHER) or a specific person at a specific time with a specific problem (NHIS).

A table of key outputs from the log file along with the same outputs from the results Postgres database are presented below to show correct flow of raw data into the final data model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Year | Data | Raw Size (rows) | Processed Size (rows) | Postgres Size (rows) |
| 2014 | Weather | 1,262,200 | 8,989 | 8242 |
| 2015 | Weather | 1,280,672 | 9,164 | 9159 |
| 2015 | NHIS | 103,789 | 5,281,356 | 5,281,356 |
| 2016 | Weather | 1,328,920 | 9,064 | 9,067 |
| 2016 | NHIS | 97,169 | 4,948,785 | 4,948,785 |
| 2017 | Weather | 1,392,311 | 9,134 | 9,137 |
| 2017 | NHIS | 78,132 | 3,978,255 | 3,978,255 |
| 2018 | Weather | 1,381,063 | 9,062 | 9,057 |
| 2018 | NHIS | 72,831 | 3,710,454 | 3,710,454 |
| 2019\* | Weather | None | None | 751 |

Table 1: Row counts for raw data, processed data, and inserted data. Note that no data was processed from 2019. Due to shifting of datetime values to create lagged observations of weather data, data collected in December, gets associated with January of the following year. This causes the per year counts in Postgres to be different from the per year count in the given weather file. The total number of processed weather rows equals the total number of rows in the WEATHER table. The row counts per year in the Postgres table can be computed using count\_nhis\_by\_year and count\_weather\_by\_year queries in sql\_queries.py

**Step 5:**

The goal of this project was to create a cleaned data model for analyzing trends in environmental air pollution, meteorological data, and self-reported health problems. The queries this data model supports allow for summarizing these trends overtime and at specific locations. Ultimately, the data model supports analyzing this time and space data to identify correlations between the fact tables, WEATHER and NHIS.

For this specific use case the Spark (or Dask) could be implemented to handle the data processing if the project were to expand. For instance, some air quality measurements were excluded because I am specifically interested in particulate matter 2.5 for this analysis. If additional types of data that are available get added to the pipeline in the future, migrating to processing libraries that can handle distributed computing would be beneficial. Besides the type of data points that could be included, data is available from additional years prior to 2014/2015. As the number of years of data increases, the need for accelerated processing would increase. I chose to use only the most recent data because 1) it speeds up development for this proof of concept project, and 2) the NHIS survey data is only available in CSV format from 2015 onwards. Data from prior to 2014 is only available as DAT files that require proprietary software to parse.

Even using this non-exhaustive dataset, I ran into problems with the amount of time it took to run the ETL job. In one step, I needed to apply a function row wise to various combinations of columns. Initially, this took to long to be feasible. To overcome this bottleneck, I was able to drop all the NULL values, apply the function to the few non-null rows, and map the results back to the original data using the index values. This strategy cut down the number of rows that needed to be processed by >99% (because most people do not have the particular health problem that is being operated on).

If I were to incorporate Airflow into this project, it would be to handle backfilling the data. The datasets used for this project get updated no more than 2-4 times per year. However, I could use Airflow to schedule the task to run once a day until all the years of historical data have been processed. This would cut down on having to run one very long batch job. Otherwise, the data need only be updated yearly moving forward, so setting up Airflow seems like overkill for this task.

I chose the tools for this project (Postgres and Python) because they work well together. Additionally, Python has lots of libraries for performing data processing. In particular, I utilized the Pandas data analysis library which made it easy to parse the data to achieve the desired data model outcome.

If the data were to increase by 100x I would 1) transition from using Pandas as the processing library to Dask, 2) store the raw data in a AWS S3 bucket, 3) move the processing to an appropriately sized AWS EC2 instance, and 4) migrate the Postgres database from my machine to AWS. Dask has an almost identical API to Pandas, which makes the transition relatively easy. Moving the data to the cloud would remove the memory constraints that exist from running the system on my local machine. Hosting the Postgres database on AWS allows for scaling out the size of the database and the number of workers that could process the data.

If the pipelines were to run on a daily basis at 7am then that would indicate that the project would have changed drastically. The data sources summarize a year of data at a time, so there is really no need to run that frequently, nor is the processing setup to handle daily data. The pipeline could be run every morning at 7am, but it would be extremely inefficient for this use case.

If the database needed to be accessed by 100+ people I would utilize a cloud service (likely AWS) to host the Postgres instance. A cloud instance of the database with enough worker nodes would allow for many users to access the data simultaneously.